**HTTP**

**Hypertext Transfer Protocol (HTTP)** is an application layer protocol for transmitting hypermedia documents, such as HTML.

**Difference Between HTTP 1.1 and HTTP 2.0**

|  |  |
| --- | --- |
| **HTTP 1.1** | **HTTP 2.0** |
| 1. HTTP/1.1 loads resources one after the other, so if one resource cannot be loaded, it blocks all the other resources behind it. 2. It sends data as plain text. 3. Client must request for Resources then only server send requested contents. 4. HTTP/1.1 is slower than HTTP/2, because it uses normal compression like gzip on HTTP message. 5. A new communication cycle is required for each resource data. 6. It is less secure than HTTP/2, because it send data in plain text. | 1. HTTP/2 is able to use a single TCP connection to send multiple streams of data at once so that no one resource blocks any other resource. 2. It split data into Binary data and arrange them before sending. 3. HTTP/2 allowing a server to "push" content to a client before the client asks for it. 4. HTTP/2 is faster than HTTP/1.1, because it uses Advance compression method known as “HPACK” on HTTP message. 5. It transmits more data per client-server communication cycle greatly improves web performance. 6. It overcomes common API security threats to protect sensitive data. |

A picture containing diagram

Description automatically generated

**OBJECTS in JavaScript**

**Objects** in JavaScript may be defined as an unordered collection of related data, of primitive or reference types, in the form of “key: value” pairs.

Syntax :

let object\_name = {

key\_name : value,

. . .

}

Example :

let person = {

firstName : "John",

lastName : "Doe",

age : 50,

displayName : function(){

console.log(`${person.firstName} ${person.lastName}`)

}

};

person.displayName();

**Ex. 1.1**

In this example “**person**” is an object. **“firstName”, “lastName” and “age”** are all **keys** and “**john**”, “**Doe**” and **50** are **values** of these **keys** respectively. Each of these keys referred to as **properties** of the object. An object in JavaScript may also have a function as a member, in which case it will be known as a **method** of that object. Here  **“displayName”** is a **method** of the person object that is being used to work with the object’s data, stored in its properties.

Objects can be created using Object() constructor or the object initializer using new Object() or Object.create() .

**JavaScript Object Properties**

Properties are the most important part of any JavaScript object.

Properties can usually be added, updated, and deleted, but some are read only.

**Accessing JavaScript Properties**

The syntax for accessing the property of an object is:

*objectName.property*// person.age //(Dot Notation)

or

*objectName*["*property*"]   // person["age"] //(Bracket Notation)

or

*objectName*[*expression*]   // x = "age"; person[x]

**Adding new Properties**

You can add new properties to an existing object by simply giving it a value.

Example :person.nationality : “Indian”; //this will add new property nationality

**Updating Properties**

You can update properties which are already present in object

Example : person.age = 30; //This will update value of age in ex. 1.1

**Deleting JavaScript Properties**

The delete keyword deletes a property from an object:

Example : delete person.age;

Or

Delete person[“age”]

The delete keyword deletes both the value of the property and the property itself.

The delete operator is designed to be used on object properties. It has no effect on variables or functions.

**Inherited Properties:**JavaScript objects inherit the properties of their prototype. The delete keyword does not delete inherited properties, but if you delete a prototype property, it will affect all objects inherited from the prototype.

**Property Attributes** Data properties in JavaScript have four attributes.

* **value:** The property’s value.
* **writable:** When true, the property’s value can be changed
* **enumerable:** When true, the property can be iterated over by “for-in” enumeration. Otherwise, the property is said to be non-enumerable.
* **configurable:**If false, attempts to delete the property, change the property to be an access-or property, or change its attributes (other than [[Value]], or changing [[Writable]] to false) will fail.

**JavaScript Object Methods**

A JavaScript **method** is a property containing a **function definition**.

Methods are functions store as object properties

**Accessing Object Methods**

**Syntax:**  ObjectName.methodName()

You will typically describe fullName() as a method of the person object, and fullName as a property.

The fullName property will execute (as a function) when it is invoked with ().

This example accesses the fullName() **method** of a person object:

**Example :**  const person = {

firstName: "John",

lastName: "Doe",

id: 5566,

fullName: function() {

return this.firstName + " " + this.lastName;

}

};

name = person.fullName();

If you access the methodName **property**, without (), it will return the **function definition**.